Java – Overview

Java was originally developed by **Sun Microsystems**, initiated by **James Gosling**, and released in **1995** as the core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

- The latest release is **Java SE 8**.
- Different configurations were introduced to cater to various platforms:
  - **J2EE**: Enterprise Applications
  - **J2ME**: Mobile Applications

These were later renamed:

- **Java SE** (Standard Edition)
- **Java EE** (Enterprise Edition)
- **Java ME** (Micro Edition)

Java's mantra: **Write Once, Run Anywhere (WORA)**.

Key Features of Java

1. **Object-Oriented**: Everything in Java is an object, supporting easy extensibility.
2. **Platform Independent**: Java compiles into platform-independent **bytecode**, which runs on any platform with a JVM.
3. **Simple**: Designed to be easy to learn, especially for those familiar with OOP.
4. **Secure**: Enables virus-free, tamper-free development using public-key encryption.
5. **Architecture-Neutral**: Compiled code runs on many processors with Java runtime.
6. **Portable**: No implementation-dependent aspects; the compiler is written in ANSI C.
7. **Robust**: Eliminates error-prone situations with extensive compile-time and runtime checks.
8. **Multithreaded**: Allows simultaneous task execution for smooth, interactive applications.
9. **Interpreted**: Bytecode is translated on-the-fly to native instructions.
10. **High Performance**: Achieved through **Just-In-Time (JIT)** compilers.
11. **Distributed**: Designed for distributed computing in the internet environment.
12. **Dynamic**: Adapts to evolving environments, with extensive runtime information.

History of Java

- **June 1991**: Initiated by James Gosling for a set-top box project.
- Initially called **'Oak'**, later renamed **'Green'**, and finally **'Java'** (chosen from a random list).
- **1995**: Sun Microsystems released Java 1.0, promising **Write Once, Run Anywhere (WORA)**.
- **2006**: Sun released much of Java as **open-source software** under the GNU General Public License (GPL).
- **2007**: The core code became entirely open-source, with a few exceptions.

Try It Option Online

You don't need to set up a local environment to start learning Java!

- Use the **Try it option** available online to compile and execute code.
- Modify and experiment with examples to enhance your understanding.

Here's an example you can try online:

```
public class MyFirstJavaProgram {
    public static void main(String []args) {
        System.out.println("Hello World");
    }
}
```

Local Environment Setup

If you prefer to set up your local environment for Java programming, follow these steps to download and configure Java on your machine:

1. **Download Java SE**
    - Java SE is available for free at the Oracle Java Downloads page.
    - Select and download the version compatible with your operating system.
2. **Install Java**
    - Run the downloaded `.exe` file and follow the installation instructions to install Java on your system.
3. **Set Environment Variables**
   After installation, set up your environment variables to point to the correct Java directories.

Setting Up the Path for Windows

Assume Java is installed in `C:\Program Files\Java\jdk` directory. Follow these steps:

1. **Right-click** on **'My Computer'** and select **'Properties'**.
2. Go to the **'Advanced' tab** and click the **'Environment Variables'** button.
3. Locate the **Path** variable and edit it:
    - Add the Java executable path to the existing value.
    - Example: If the current path is `C:\WINDOWS\SYSTEM32`, update it to:

```
C:\WINDOWS\SYSTEM32;C:\Program Files\Java\jdk\bin
```

Setting Up the Path for Linux/UNIX/Solaris/FreeBSD

1. Set the `PATH` environment variable to include the Java binaries directory.
2. Edit your shell configuration file (e.g., `.bashrc` for bash shell).

3. Add the following line at the end of the file:

```
export PATH=/path/to/java:$PATH
```

Replace `/path/to/java` with the actual path to the Java installation directory.

Popular Java Editors

To write Java programs, you need a text editor or an Integrated Development Environment (IDE). Below are some popular options:

1. **Notepad**:
    - Simple text editor available on Windows.
    - Recommended for this tutorial.
2. **NetBeans**:
    - Open-source and free Java IDE.
    - Download from: NetBeans.
3. **Eclipse**:
    - Developed by the Eclipse open-source community.
    - Download from: Eclipse.

Here's the organized and structured content for **Java - Basic Syntax**:

3.0 Java – Basic Syntax

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let's understand the core concepts:

Key Concepts:

- **Object**: Objects have states and behaviors. Example: A dog has states like color, name, breed, and behaviors like wagging its tail, barking, and eating. An object is an instance of a class.
- **Class**: A class is a template or blueprint that describes the behavior or state an object of its type will have.
- **Methods**: A method represents behavior. A class can contain many methods where the logic is written, data is manipulated, and actions are executed.
- **Instance Variables**: Each object has its own set of instance variables, and its state is defined by the values assigned to these variables.

First Java Program

Let's write a simple program to print "Hello World":

```java
public class MyFirstJavaProgram {
    /* This is my first Java program.
    * This will print 'Hello World' as the output.
    */
    public static void main(String[] args) {
        System.out.println("Hello World"); // prints Hello World
    }
}
```

Steps to Save, Compile, and Run the Program:

1. **Open Notepad** and add the code above.
2. **Save the file** as `MyFirstJavaProgram.java`.
3. Open the **command prompt** and navigate to the directory where you saved the file. Assume it's `C:\`.
4. **Type** `javac MyFirstJavaProgram.java` and press **Enter** to compile your code. If there are no errors, the command prompt will return to the next line.
5. **Type** `java MyFirstJavaProgram` to run the program.
6. You should see `Hello World` printed in the window.

Example:

```
C:\> javac MyFirstJavaProgram.java
C:\> java MyFirstJavaProgram
Hello World
```

Basic Syntax

Important points to keep in mind when writing Java programs:

- **Case Sensitivity**: Java is case-sensitive, meaning `Hello` and `hello` are considered different identifiers.
- **Class Names**: Class names should begin with an uppercase letter. If there are multiple words, each inner word's first letter should be capitalized.
  - Example: `class MyFirstJavaClass`
- **Method Names**: Method names should start with a lowercase letter. Similarly, inner words should have their first letter capitalized.
  - Example: `public void myMethodName()`
- **Program File Name**: The file name must exactly match the class name, with a `.java` extension.
  - Example: For class `MyFirstJavaProgram`, the file should be saved as `MyFirstJavaProgram.java`.
- **Main Method**: Every Java program must have a `main()` method to begin processing.
  - Example: `public static void main(String args[])`

Java Identifiers

Identifiers are names used for classes, methods, and variables in Java. Points to remember about identifiers:

- Identifiers must begin with a letter (A-Z or a-z), dollar sign (`$`), or an underscore (`_`).
- After the first character, identifiers can contain any combination of letters, digits, dollar signs, and underscores.
- Reserved keywords cannot be used as identifiers.
- Java identifiers are **case-sensitive**.

Examples:

- **Legal Identifiers**: `age`, `$salary`, `_value`, `__1_value`.
- **Illegal Identifiers**: `123abc`, `-salary`.

Java Modifiers

Modifiers are used to alter the behavior of classes, methods, and other entities in Java. There are two types of modifiers:

1. **Access Modifiers**:
    - `default`, `public`, `protected`, `private`
2. **Non-access Modifiers**:
    - `final`, `abstract`, `strictfp`

(Note: We will explore modifiers in more detail in the next section.)

Java Variables

There are three types of variables in Java:

1. **Local Variables**: Defined inside methods and accessible only within that method.
2. **Class Variables (Static Variables)**: Defined with the `static` keyword, shared among all instances of the class.
3. **Instance Variables (Non-static Variables)**: Defined without the `static` keyword, unique to each object.

Java Arrays

Arrays are objects that store multiple values of the same type. An array itself is an object on the heap, and we will cover how to declare, construct, and initialize arrays in upcoming chapters.

Java Enums

Enums, introduced in Java 5.0, restrict a variable to have one of only a few predefined values. This helps reduce bugs in the code.

Example: Fresh Juice Shop Enum

```
class FreshJuice {
    enum FreshJuiceSize { SMALL, MEDIUM, LARGE }
    FreshJuiceSize size;
}

public class FreshJuiceTest {
    public static void main(String[] args) {
        FreshJuice juice = new FreshJuice();
        juice.size = FreshJuice.FreshJuiceSize.MEDIUM;
        System.out.println("Size: " + juice.size);
    }
}
```

Output:

```
Size: MEDIUM
```

Enums can be declared inside or outside a class. Methods, variables, and constructors can also be defined within enums.

Java Keywords

Java has reserved words, called keywords, that cannot be used as identifiers. Below is the list of all Java keywords:

- `abstract`, `assert`, `boolean`, `break`, `byte`, `case`, `catch`, `char`
- `class`, `const`, `continue`, `default`, `do`, `double`, `else`, `enum`
- `extends`, `final`, `finally`, `float`, `for`, `goto`, `if`, `implements`
- `import`, `instanceof`, `int`, `interface`, `long`, `native`, `new`, `package`
- `private`, `protected`, `public`, `return`, `short`, `static`, `strictfp`, `super`
- `switch`, `synchronized`, `this`, `throw`, `throws`, `transient`, `try`, `void`
- `volatile`, `while`

Comments in Java

Java supports two types of comments, which are ignored by the Java compiler:

1. **Single-line comments**:
    - Denoted by `//`. Everything after `//` on that line is ignored by the compiler.
2. **Multi-line comments**:
    - Denoted by `/* */`. Everything between `/*` and `*/` is ignored.

Example:

```java
public class MyFirstJavaProgram {
    /* This is my first Java program.
     * This will print 'Hello World' as the output.
     * This is an example of multi-line comments.
     */
    public static void main(String[] args) {
        // This is an example of a single-line comment.
        /* This is also an example of a single-line comment. */
        System.out.println("Hello World");
    }
}
```

Using Blank Lines

A line containing only whitespace or a comment is considered a **blank line**. Java ignores blank lines.

Inheritance

In Java, you can derive a new class from an existing class. This allows you to reuse the fields and methods of the existing class, known as the **superclass**, without rewriting code. The new class is called the **subclass**.

Interfaces

An interface defines a contract between objects on how to communicate. It specifies methods that the subclass must implement, but the actual implementation is up to the subclass.